BOLT BERANEK AND NEWMAN INC
CONSULTING · DEVELOPMENT · RESEARCH

Report No. 2186
Job No.s 11430
11545

AFOSR - TR - 71-2790

INFORMATION PROCESSING MODELS AND COMPUTER AIDS FOR HUMAN PERFORMANCE

FINAL REPORT, SECTION 2
Task 2: MODELS OF HUMAN-COMPUTER INTERACTION

30 June 1971

ARPA ORDER No. 890, Amendment No. 5
Sponsored by the Advanced Research
Department of Defense, under Air Force Office
of Scientific Research Contract F44620-67-C-0033

Prepared for:
Air Force Office of Scientific Research
1400 Wilson Boulevard
Arlington, Virginia 22209

24

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Bolt, Beranek and Newman Inc. 50 Moulton Street Cambridge, Massachusetts 02138	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE INFORMATION PROCESSING MODELS AND COMPUTER AIDS FOR HUMAN PERFORMANCE TASK 2: MODELS OF HUMAN-COMPUTER INTERACTION

4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Scientific Final

5. AUTHOR(S) (First name, middle initial, last name)

Mario C. Grignetti

Duncan C. Miller

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
30 June 1971	21	0

8a. CONTRACT OR GRANT NO. F44620-67-C-0033	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. 9890	
c. 61101D	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AFOSR - TR - 71 - 2790
d. 681313	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES TECH, OTHER	12. SPONSORING MILITARY ACTIVITY AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (NL) 1400 WILSON BLVD ARLINGTON, VIRGINIA 22209

13. ABSTRACT

We have conducted experiments to explore methods of motivating time-sharing users to adopt behavior patterns that improve overall system performance. It was found that while it is indeed possible for a time-sharing system to provide incentives to users that will affect their choices between alternative methods of accomplishing a given task, the extent of this effect is not entirely predictable.

We have also designed and implemented a measuring system for the SDS-940 time-sharing computer system. This measuring system yielded data that were useful in increasing our understanding of the dynamic behavior of programs in a time-sharing system and, more specifically, in improving overall system performance.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

INFORMATION PROCESSING MODELS AND COMPUTER AIDS FOR HUMAN PERFORMANCE

FINAL REPORT, SECTION 2

Task 2: MODELS OF HUMAN-COMPUTER INTERACTION

30 June 1971

by

Mario C. Grignetti
Duncan C. Miller

ARPA Order No. 890, Amendment No. 5
Sponsored by the Advanced Research Projects Agency, Department of Defense, under Air Force Office of Scientific Research Contract F44620-67-C-0033

Prepared for

Air Force Office of Scientific Research
1400 Wilson Boulevard
Arlington, Virginia 22209

TABLE OF CONTENTS

FINAL TECHNICAL REPORT

ARPA Order No. 890, Amendment No. 5
Program Code No. 9D20
Contractor: Bolt Beranek and Newman Inc.
Effective Date of Contract: 1 November 1966
Contract Expiration Date: 30 June 1971
Amount of Contract: $804,896.00
Contract No. F44620-67-C-0033
Principal Investigators: John A. Swets
Mario C. Grignetti
Wallace Feurzeig
M. Ross Quillian
Telephone No. 617-491-1850
Title: INFORMATION PROCESSING MODELS AND
COMPUTER AIDS FOR HUMAN PERFORMANCE

TASK 2: MODELS OF HUMAN-COMPUTER INTERACTIONS

1. Technical Problem

The purpose of this research program is to develop models for certain types of human-computer interactions in a time-sharing environment at the human-computer interface level.

2. General Methodology

Laboratory experiments.

3. Technical Results

We have conducted experiments to explore methods of motivating time-sharing users to adopt behavior patterns that improve overall system performance. It was found that while it is indeed possible for a time-sharing system to provide incentives to users that will affect their choices between alternative methods of accomplishing a given task, the extent of this effect is not entirely predictable.

We have also designed and implemented a measuring system for the SDS-940 time-sharing computer system. This measuring system yielded data that were useful in increasing our understanding of the dynamic behavior of programs in a time-sharing system and, more specifically, in improving overall system performance.

4. Department of Defense Implications

Large savings in the cost of software development are potentially possible by converting from the batch-processing computer systems that are widely used today to interactive, time-shared computer systems. To design, operate, or even select such interactive systems in a rational way, it is necessary to be able to predict its relative acceptability, i.e., how users will behave with a system having given response characteristics.

5. Reports Annotated Within

Grignetti, M. C. and Miller, D. C. "Modifying Computer Response Characteristics to Influence Command Choice," Proceedings of the IEE Conference on Man-Computer Interaction, Publication No. 68, September 1970, 201-206.

1. PREFACE

At its inception in 1966, this contract was devoted solely to the one area of second-language learning. Later amendments have added three more tasks: Models of Man-Computer Interaction; Programming Languages as a Tool for Cognitive Research; and Studies of Human Memory and Language Processing. The present contract was scheduled for termination on 31 December 1970, but the final reporting date was changed to 30 June 1971, to allow completion of data analysis in the various tasks.

Due to the amount of information to be presented in the Final Report, we have bound it in four Sections, one for each task. In addition to a copy of this page, each Section contains an appropriate subset of the documentation data required for the report: a contract-information page, a summary sheet for the particular task at hand, and a DD form 1473 for document control.

2. ANNOTATED BIBLIOGRAPHY

Grignetti, M. C. and Miller, D. C. "Modifying Computer Response Characteristics to Influence Command Choice," Proceedings of the IEE Conference on Man-Computer Interaction, September 1970, Publication No. 68, 201-206.

This paper summarizes the results and conclusions reported in detail in our Semiannual Technical Report No. 7, in which we describe the work performed and the results obtained from two series of human-computer interaction experiments. These experiments were designed to test the feasibility of methods for improving the overall efficiency of a time-sharing system (that is, the efficiency of the system and its users, considered together), by artificially manipulating the computer's response characteristics so as to influence the user's choice of interaction commands. The results have demonstrated that it is indeed possible for a time-sharing system to provide incentives to users that will cause them to modify their behavior in the desired way. However, the extent of this behavior modification is not exactly predictable without detailed knowledge of the particular circumstances and the prejudices of the users.

3. OVERVIEW

We are concerned with situations involving user-computer interactions in a time-sharing system environment and with the relationship between user behavior and overall system performance.

We aim at developing quantitative models for the dynamic behavior or user-computer systems. In particular, we are interested in models that describe the dynamic nature of the service demands that users make on time-shared computer systems, as well as in the time-sharing system's behavior in response to these demands. Such models are important for the design, analysis, and evaluation of user-computer systems.

3.1 User Modelling

One of our tasks under this contract has been to find ways to influence user behavior via modification of the computer response characteristics, so that system performance is improved. To this end, we designed experiments in which a text-editing task was performed by practiced subjects on a time-sharing system. The subjects were allowed to choose among alternative methods of correcting errors, which required a trade-off between the subject's planning time and the rate at which computer resources were demanded. A "cost" that represented this demand rate was deducted from the subject's incentive pay after each command and was fed back to the subject as a part of the computer's response.

It was found that it is possible for a time-sharing system to provide incentives to users that will affect their choices between alternative methods of accomplishing a task. However, the extent of this effect is not precisely predictable.

3.2 Computer Modelling

During the period covered by this report there have been two computer systems that provided the basic time-sharing environment for our research. From the time of inception of the contract until June 1970 we had available an SDS-940 time-sharing system, and from July 1970 to the end of the reporting period, December 1970, we based our work on the TENEX* system operating on a modified DEC-PDP-10 computer. Due to the limited time we have had available to work with the TENEX system, most of our detailed computer modelling work was done on the SDS-940 system.

The model consists of a network of Queueing Theory servers and a set of user processes that circulate among them as units. Processes are run one at a time by the RUN server until a termination condition is reached. The termination conditions are:

a) The process requires I/O transfer of data.
b) Time quantum has been exceeded.
c) Process has dismissed itself

When a process becomes runnable again, (for example, after the I/O transfer has been completed, or immediately after a quantum overflow), it waits in a multi-level queueing structure characterized by order of priorities and queue discipline. The highest priority is granted to processes that have finished inputting certain kinds of data via the controlling teletype, and the lowest priority to processes that have exhausted their long quantum or that require relatively long I/O transfer times. In the latter case, the core memory assigned to the process becomes eligible for running other higher priority processes and drum swaps may be necessary.

*TENEX is an operating system developed with the joint support of BBN and of the Advanced Research Projects Agency of the DOD.

The task of building a model for the dynamic behavior of a time-sharing system is a highly interactive one.

First, the executive program or monitor must be studied and understood, so that its states can be identified. These states are characterized by the fact that user processes enter and leave them in a well defined manner. Thus, teletype input is a state because a user process enters it whenever teletype character input is demanded, and leaves it when either a terminating character is typed in or the input buffer fills up. Conversely, scheduling is not a state because user processes do not enter and leave the scheduler, but rather are manipulated and assigned to states by it.

Once a preliminary set of states and the transitions from one state to another are identified, some experimental data must be gathered not only to quantify the model but also to check it qualitatively.

To this end, we implemented a software Measuring System that allowed us to obtain the following information:

a) The probability densities of time spent in the different states, i.e., running processes, transferring information to the various I/O devices, waiting at the various queue levels, and being swapped to and from the drum.

b) The transition probabilities from one state to another.

c) The probability density for the number of processes in each state.

d) The probability density for the number of pages being transferred to and from the drum during a process swap.

The Measuring System worked by gathering the necessary information with a minimum of processing. Small patches introduced into the Monitor obtained the three words of data that were necessary to define each event, and a logging subroutine stored this information in a ring buffer. A user process was activated before the buffer filled up, and the buffer contents were dumped into a disk file. In this way a sequential data store was kept, and time history information was preserved.

After the data were collected, a Data Reduction program was activated. This program analyzed the information stored and produced the histograms needed to estimate the various probability densities. A third program evaluated and presented the results in tabular form.

Consider the following example as an illustration of one of the useful applications of the Measuring System. An early implementation of MINITECO failed to perform as expected with regard to response time: when the time-sharing system was operating under medium-load conditions, the response times we obtained were considerably longer than planned. To be able to understand what was slowing down the execution of MINITECO commands, we used our Measuring System at a time when the only users present in the system were our experimental subjects.

With the response time fixed at 3 seconds we observed that the probability density of Run time for each burst of execution had a mode of 2.5 milliseconds, a median of 6 milliseconds, and a mean of 25 milliseconds, the latter value due to the effect of a few very long bursts. The short quantum in the system was 160 milliseconds, so it was apparent that the program was not using CPU time effectively. The data revealed also that most of these short Runs were due to requests for transfer of information to and from the Drum, the average transfer requiring about 20 milliseconds for completion. The picture that emerged from this was that response time characteristics should improve if the program were made to make better use of in-core information, so as to reduce the number of drum transfers required per program execution.

After appropriate modifications were made, the response time was drastically reduced and became almost solely dependent on teletype output. That is, the response time was reduced essentially to the time required to type out the computer's response to the user's command.

Another useful consequence of the use of the Measuring System was to make clear the importance of "dynamic bugs," Most programs are debugged following an essentially static approach; i.e., given certain initial conditions, the programmer knows what the state of the machine ought to be at each of a set of check points in the course of executing the program. The program is considered statically debugged when the actual states coincide with the expected states at each check point. But in highly complex interactive systems, this view is no longer sufficient, because it might happen that although the machine, starting from checkpoint A indeed arrives at point B with the right state, it may actually execute the program along trajectories entirely unacoounted for, and undetectable by the static debugging approach. This may lead to inefficiency in the computer's operations, as the following two examples demonstrate.

a) When the system was heavily loaded, we expected to see the number of user processes requesting drum swaps increase. Actual measurement revealed that there was never more than one process in this condition. Closer inspection of the Monitor Code revealed a bug which essentially prevented a process from requesting a swap if another process was being swapped. Thus, the sophisticated software that was supposed to handle multiple swaps had never been used!

b) When a user process requested a disk file transfer, the disk directory had to be locked to prevent other processes from using it. The scheduler was supposed to notice this condition and not to select for running any processes that were waiting for the disk to be free. The measuring system revealed, however, a surprisingly high number of short runs terminating in disk file transfer requests. Again, closer inspection revealed that the scheduler was not performing the right test and allowed processes that were waiting for the disk to begin running in spite of the disk's being busy. This run time was being entirely wasted, since when the processes found the disk still busy, they had to be dismissed again.

Notice that neither of these dynamic bugs, nor the anomalous behavior of user processes that they induced, could have been caught by the static debugging approach. In fact, the system appeared to be running quite satisfactorily, until the Measuring System revealed these dynamic bugs. Once they were corrected, however, the system efficiency increased—the drum was utilized as designed, and wasteful runs were eliminated.

4. REPORTS

The paper annotated in Section 2 is included in this report immediately after this page.

MODIFYING COMPUTER RESPONSE CHARACTERISTICS TO INFLUENCE COMMAND CHOICE*

Mario C. Grignetti and Duncan C. Miller

Bolt Beranek and Newman Inc.
Cambridge, Massachusetts 02138

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F44620-67-C-0033

10

Introduction

The purpose of this investigation is to explore methods of motivating time-sharing system (TTS) users to adopt behavior patterns that improve overall system performance.

User behavior and system performance are interdependent: system response time depends upon the number of users and the operations they are conducting, while the commands chosen by the users often depend upon the apparent system responses. Generally, a user can choose among several alternative sets of commands that will accomplish a certain goal. Other things being equal, he will choose a simple command over a complex one, a rapidly executed command over a time-consuming one, or, if the difference is apparent to him, a "cheap" command over an "expensive" one.

To make a more sophisticated choice on the basis of maximizing total system efficiency (and hence the long run benefit to all users), he would need a fairly detailed knowledge of the system dynamics and the current demands of other users. The latter conditions prevail only in closely knit research computer installations where the users know each other well and can coordinate their activities by direct interpersonal communication. In large, remote access TSS's this is virtually impossible. Hence, the TSS itself must provide the means to coordinate and regulate user behavior.

One way to do this would be to incorporate into the TSS the capability of providing incentives to lead individual users to adopt behavior which, although it may seem against his best interests at first sight, will result in his greatest satisfaction in the long run, and which will optimize overall system performance.

What might these incentives be? One system characteristic that affects user behavior is the apparent system response time. Presume that (as in generally the case) a user may choose among several different series of commands to achieve a certain goal. Some series of commands will require a low rate of expenditure of the computer's resources, but will require careful on-line planning. Others may require expenditure of the computer's resources at a greater rate, but will demand much less planning. Which will he choose? His choice will depend on the tradeoff he perceives between his planning effort and the system response time. If the system is lightly loaded and responds quickly to any series of commands, he will probably choose the series that minimizes his planning effort. If, however, the system responds sufficiently faster to a well-planned series of commands, then he will find the extra planning effort worthwhile. If a system designer could predict the user's choices, then he might attempt to discourage operations that result in inefficient system performance by placing an artificial time penalty on such operations.

We suggest, however, that there are other ways to affect a user's behavior without inflicting artificial time penalties upon him. Some approximation to the real "cost" of a command (in terms of its load on system resources) could be made explicitly available to him, and he could be encouraged to balance the "cost" of various commands against the planning and execution times that they require. For example, a user might be allotted a certain number of cost units as he begins a session. He could receive high priority service until he used up the allotted cost units; then he would receive some lower priority service. This would encourage him to weigh carefully the costs of alternative commands against the planning times required.

We have conducted experiments to discover to what extent users may reasonably be expected to optimize their decisions and behave uniformly and predictably when cost and time information are explicitly provided to them by the computer system. These experiments were conducted in a sufficiently constrained way that the optimality of the users' decisions could be evaluated easily. These constraints were necessary to allow us to analyze the many factors involved in a user's decisions. At the same time, however, we attempted to keep our experiments sufficiently representative of real-life tasks that the techniques we developed for predicting and modifying user behavior could be applicable to real-world computer systems and their varying populations of users.

The Experimental Task

The task we chose involved correcting typographical errors introduced into fixed-syntax sentences generated by selecting at random an article, an adjective, a noun, and so on. An example is:

THE HIRSUTE PORCUPINE ANGRILY PUNTED A CRUMPLED SURFBOARD

The errors introduced into each page of 100 such sentences were carefully selected to keep the task difficulty constant for each page. Among the error parameters controlled were the length of the sentence, the position of the error in the sentence, the minimum number of characters necessary to specify uniquely the position of the error, the number of characters be deleted, and the number of characters to be inserted. The generation of errors was automated to provide a virtually inexhaustible supply of error text.

An editing program (MINITECO) was written that provided our subjects with three distinct methods of correcting an error:

(1) The KILL command erased a sentence and allowed the subject to retype the correct version in its entirety.

(2) The DELETE/INSERT command required the subject to count the number of characters up to the error and to input this number, the number of characters to be deleted (if any), and the characters to be inserted (if any).

(3) The REPLACE command was of the form "replace 'old string' with 'new string', where 'old string' includes the error plus any preceding characters that may be necessary to specify uniquely the position of the error, and where 'new string' is the corrected version of 'old string'.

After the subject entered each command, MINITECO typed out h how long it took to enter the command, issued a reward if the sentence had been corrected properly, subtracted the cost of the command used, and typed out a summary of the total amount earned and the total time used. The apparent computer response time (the time before MINITECO was ready to accept another command) was under our control, along with the costs charged for each command type and the total time per session.

Our subjects were three secretaries, all of whom were experienced typists and had some experience in editing tasks in a TSS environment. They were actually paid according to the total earnings reported by MINITECO. Since each experimental session was of fixed length, they were highly motivated to choose the command for each sentence that maximized their earnings per unit-time. The purpose of the first (no-choice) experiment was to gather data from which we could formulate input models for each command type. These models would allow us to predict how long it would take the subjects to correct a certain sentence and, in our second (choice) experiment, to determine whether our subjects were choosing properly the command types that maximized their earnings.

No-Choice Experiments

The subjects were thoroughly trained in the use of each command type, and input models were calculated for each subject using each command type at three values of computer response time (3, 9, and 27 seconds). During this phase of the experiments, each subject ran approximately 20 one-hour sessions. At the beginning of each session, the subjects were told which command type to use for each run. No hoice between commands was allowed.

The results indicated that for the shortest response time (3 seconds), the time necessary for a subject to correct an error is linearly related to the sentence length when he uses a KILL command; is linearly related to the position of the error in the sentence when he uses a DELETE/INSERT command; and is essentially constant and independent of any identifiable error parameters when he uses a REPLACE command. The results for the REPLACE command were somewhat surprising to us, as we expected to see a distinct correlation between execution time and the lenght of the minimum string of characters necessary to specify the error location. Apparently, the subjects perceived short strings of characters as units when scanning the sentence rather than as individual characters.

At the longest response time (27 seconds), the times necessary to execute a command reduced to the times necessary to type in the command string, since all planning of the command could be done while waiting for the system to carry out the previous command.

At the intermediate response time (9 seconds) a combination of these effects was seen. The time required to execute a KILL command remained proportional to sentence length. The time required to execute a REPLACE command remained essentially constant. The time required to execute a DELETE/INSERT command was constant when the error lay in the first half of the sentence and linearly related to the error position when it lay in the last half. When the error was early, the subjects could complete the counting of characters and the planning of the DELETE/INSERT command while waiting for the computer to carry out the previous command. When the error was late, they could not.

Choice Experiments

The input models indicated that if the costs of the three commands were set equal, a subject striving to maximize his pay per unit time would never use the KILL command and would use the DELETE/INSERT command only for errors very near the beginning of a sentence. In addition, the subjects indicated that they found the REPLACE command distinctly preferable to the other two.

To test whether the subjects could be motivated to modify their normal behavior, a differential cost structure was established. The subjects were offered a 10¢ reward for correcting each error, from which was subtracted 1¢ for a KILL command, 5¢ for a DELETE/INSERT command, or 6¢ for a REPLACE command. This cost structure was specifically designed to counteract the subjects reluctance to use the KILL command. It was explained to the subjects that to maximize their earnings, they would have to weight carefully the cost against the time consumed for each command for each sentence. No specific methods for doing this were suggested.

A series of 32 runs of 1000 seconds duration was conducted for each subject with a 3-second computer response time. The subjects rapidly settled upon a consistent choice strategy, which differed very little between subjects. This strategy may be roughly summarized as, "If the sentence is less than 40 characters long, use KILL. If not, and the error lies before the twentieth character, use DELETE/INSERT. Otherwise, use REPLACE."

To test the optimality of this strategy, we recomputed input models for the choice experiment data. We expected that the necessity of choosing between alternative commands would increase somewhat the execution times for each command. Instead we found that both the slope and the intercept of the best-fit linear regression models decreased slightly for KILL and DELETE/INSERT. The time required for REPLACE remained unchanged. Evidently, the subjects were able to develop more efficient techniques for using KILL on short sentences and DELETE/INSERT on early errors during the course of the experiments.

Using the recomputed input models, we discovered that the subjects' strategies were not optimal with respect to the given cost structure. The "best choice" command was used only about 50% of the time. The remaining choices were almost entirely "second best"; "third best" choices were reare. It is apparent that the subjects had been motivated to use KILL and DELETE/INSERT far more often than they would have without a differential cost structure. The strategies they adopted were consistent, and were a close approximation to the *form* of the optimal strategy. The optimum strategy, however, demanded the use of KILL for much longer sentences, and rearely required the use of REPLACE.

The similarity of the form of the subjects' strategies to that of the optimal strategy raised the possibility that the subjects had based their strategies on some set of *perceived* costs that were different from the given costs. Further analysis on the data showed that the subjects' strategies were indeed nearly optimal for a KILL cost of 4¢ rather than 1¢. With the single change, their choices became "best" over 80% of the time, with "second best" choices occurring primarily when the pay rate difference between commands was very small. The subjects apparently used KILL only when the pay rate (based on the given costs) was substantially higher than the pay rates of other commands.

Conclusions

These experiments have demonstrated that it is possible for a time-sharing system to provide incentives to users that will affect their choices between alternative methods of accomplishing a task.

On the other hand, the results indicate that even with very explicit incentives and feedback of results, users cannot be expected to overcome completely their preferences and prejudices among the alternatives. The assumption that users, given adequate incentives and information, will make optimal choices, does not appear to be generally true. Adding incentives to a time-sharing system will cause users to modify their behavior to some extent in the desired way. To what extent is not predictable without detailed knowledge of the particular circumstances and the prejudices of the users.